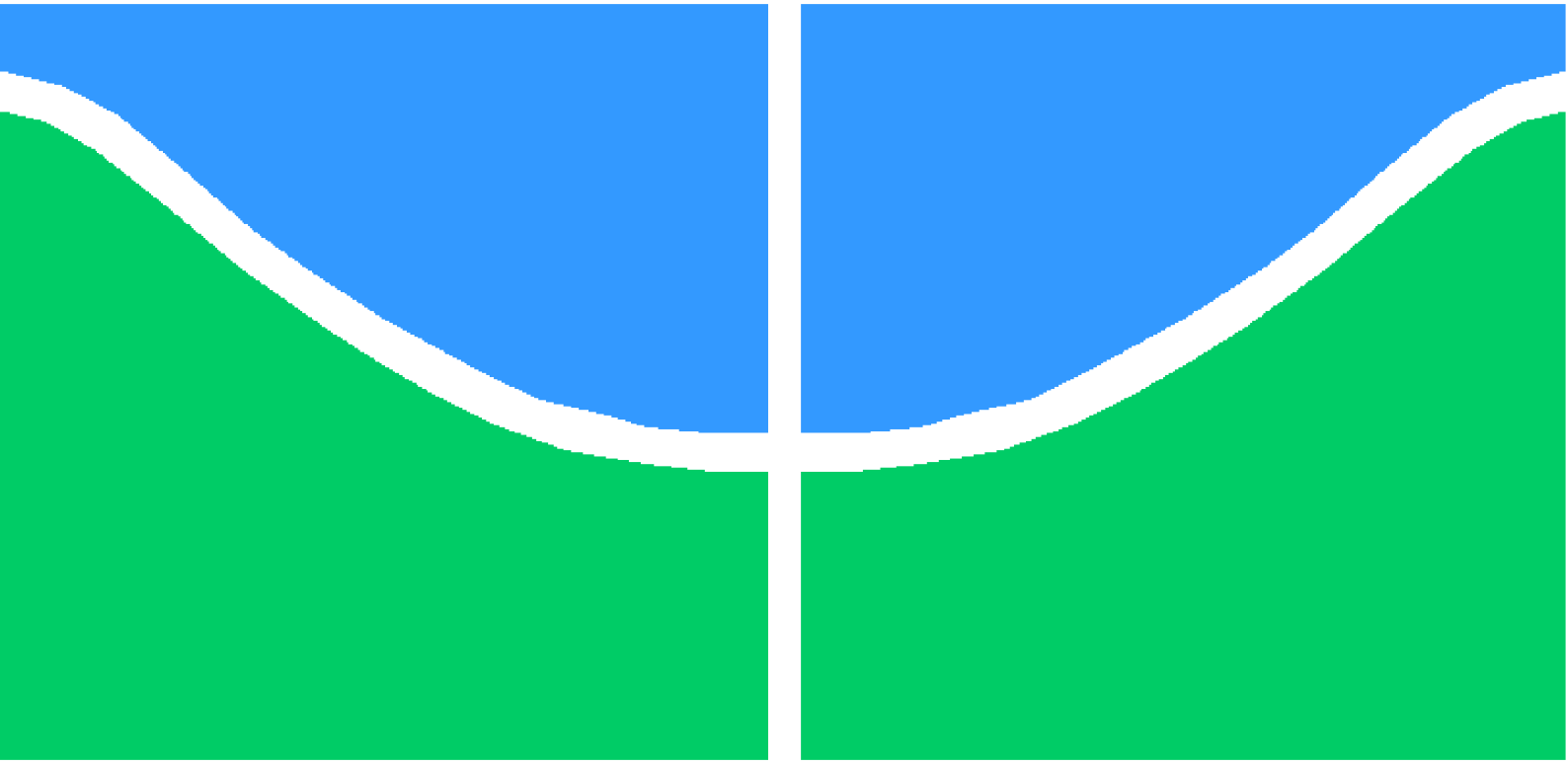

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Os 5 Desprezíveis: Desenvolvimento de um Jogo Eletrônico Utilizando os Princípios de Engenharia de Software

Autor: Eusyar Alves de Carvalho

Orientador: Dr. Edson Alves da Costa Júnior

Brasília, DF

2013

Eusyar Alves de Carvalho

Os 5 Desprezíveis: Desenvolvimento de um Jogo Eletrônico Utilizando os Princípios de Engenharia de Software

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dr. Edson Alves da Costa Júnior

Brasília, DF

2013

Eusyar Alves de Carvalho

Os 5 Desprezíveis: Desenvolvimento de um Jogo Eletrônico Utilizando os Princípios de Engenharia de Software/ Eusyar Alves de Carvalho. – Brasília, DF, 2013-
61 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Edson Alves da Costa Júnior

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2013.

1. Jogo. 2. Software. I. Dr. Edson Alves da Costa Júnior. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Os 5 Desprezíveis: Desenvolvimento de um Jogo Eletrônico Utilizando os Princípios de Engenharia de Software

CDU 02:141:005.6

Eusyar Alves de Carvalho

Os 5 Desprezíveis: Desenvolvimento de um Jogo Eletrônico Utilizando os Princípios de Engenharia de Software

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 09 de dezembro de 2013:

Dr. Edson Alves da Costa Júnior
Orientador

MsC. Hilmer Rodrigues Neri
Convidado 1

Dr. Carla Silva Rocha Aguiar
Convidado 2

Brasília, DF
2013

À todos bons e dedicados professores da minha vida.

Agradecimentos

Agradeço a Deus, por tudo.

Agradeço o apoio dos meus familiares, que sempre acreditaram e ainda acreditam em mim. Aos meus pais que sempre foram exemplo de luta e superação. Aos meus irmãos, que são fontes de exemplo em tudo o que fazem, muito obrigado.

Agradeço a cada um dos meus amigos, peças fundamentais para a conclusão desse trabalho. Sem eles eu não chegaria tão longe. Não é possível citar todos, mas agradeço de maneira especial ao senhores Greg Ouyama, Ricardo Junior, Marcus e Pablo Henrique, sem eles tenho certeza que não continuaria o meu curso.

Obrigado a todos que de forma direta ou indireta me ajudaram.

Muito obrigado.

"Nada há como começar para ver como é árduo concluir."
(Victor Hugo)

Resumo

Esse trabalho passa por todo o ciclo de vida de um jogo, desde a pré-produção até a pós-produção, abordando características especificas, tais como o desenvolvimento de conteúdo artístico e o desenvolvimento do *game design*. A experiência a ser relatada leva em consideração a aplicação de Engenharia de Software nesse processo, através do levantamento de metodologias traduzidas para contexto de jogos e de ferramentas específicas. São apresentados os dois documentos de *game design* desenvolvidos para o jogo e o processo de escolhas que levaram a conclusão e a construção de um jogo viável. Junto com esses documentos, as ferramentas utilizadas para a codificação e prototipação também são mostradas, com as suas vantagens e desvantagens verificadas ao longo do desenvolvimento. Por fim, os principais erros do desenvolvimento e os seus resultados no jogo concluído são apresentados como uma forma de aprendizado e novas áreas de pesquisa são sugeridas para trabalhos futuros.

Palavras-chaves: jogo. software. produção.

Abstract

This paper go through the entire life cycle of a game, from pre-production to post-production, addressing specific features, such as the development of artistic content and the development of *game design*. The experiment to be reported takes into account the application of Software Engineering in the process by surveying methodologies translated into the context of games and specific tools. Both *game design* documents developed for the game and the choices that led to the completion process and the construction of a viable game are presented. Along with these documents, the tools used for coding and prototyping are also shown, with their advantages and disadvantages verified throughout the development. Finally, the main errors of development and its results are presented in the game finished as a form of learning and new research areas are suggested for future work.

Key-words: game. software. production.

Lista de ilustrações

Lista de tabelas

Sumário

Introdução

Jogos estão presentes no cotidiano humano há milhares de anos e em diversas culturas. Sempre desempenhando um papel relacionado ao entretenimento e ao aprendizado, os jogos se apresentaram de diversas formas ao longo da história: cartas, tabuleiros, dados e outros (SQUIRE, 2003). Um formato recente é o eletrônico, onde o jogador interage com um aparelho eletrônico ao invés de uma ferramenta mecânica. Essa nova modalidade de jogos traz um novo leque de estudos e oportunidades.

Atualmente, não é incomum uma pessoa ter contato com pelo menos um jogo eletrônico ao longo do seu dia. Desde o mais simples console doméstico até os mais avançados *smartphones* e televisores, os aplicativos de jogos estão presentes no cotidiano de famílias e até mesmo em comerciais e campanhas publicitárias. Apenas em 2012, nos Estados Unidos, as vendas relacionadas a jogos eletrônicos chegaram a 20 bilhões de dólares (ASSOCIATION et al., 2013).

A forma mais comum de um jogo eletrônico é um *software* instalado em algum tipo de computador. Uma vez que o jogo eletrônico não é nada mais que um *software* voltado ao entretenimento, o desenvolvimento de um jogo eletrônico está intrinsecamente ligado ao desenvolvimento de *software*.

No entanto, a área de entretenimento traz consigo uma nova gama de dificuldades e obstáculos que devem ser superados para se alcançar um jogo de qualidade e sucesso. Dentre as principais características do desenvolvimento de jogos eletrônicos está a necessidade de mudanças constantes nos requisitos e a multidisciplinaridade da equipe (SANTOS, 2006), sempre cruciais e constantes.

A mudança dos requisitos é uma reflexão do dinamismo necessário para o posicionamento no mercado, conhecido por modas e questões subjetivas, que toda empresa da área deve ser capaz de se adaptar. É comum que diversas características sejam descartadas depois de terem sido implementadas, visando uma jogabilidade (*gameplay*) mais divertida. Essa explosão de criatividade pode se tornar uma grande inimiga durante a produção (BETHKE, 2003), sendo que por muitas vezes acaba por se tornar um fator de insucesso.

Já a multidisciplinaridade da equipe é consequência da alta complexidade que envolve cada projeto de jogo. Muitos métodos tradicionais, utilizados nas mais diversas áreas de conhecimento relacionados a Engenharia de Software, não podem ser diretamente aplicados nesse novo contexto (PETRILLO et al., 2008). Para esses casos, é necessário encontrar metodologias capazes de executar a mesma tarefa e que sejam de conhecimento amigável o suficiente para que toda a equipe possa exercer a metodologia.

Inspirado pelo mercado brasileiro e pelo seu constante crescimento (ABRAGAMES, 2008), esse trabalho procura unir algumas pesquisas realizadas sobre *game design* e produção de jogos com os conceitos existentes das principais areas de Engenharia de Software de forma a construir um jogo do início ao fim, chamado de ***Os 5 Desprezíveis***, um jogo de detetive *single player*. Serão documentados as suas principais falhas e oportunidades de melhoria no desenvolvimento. Através da produção do ***Os 5 Desprezíveis***, os conceitos relacionados a Engenharia de Software serão exercitados.

0.1 Organização do Trabalho

Este trabalho está dividido em capítulos. O primeiro trata da fundamentação teórica, onde conceitos relativos a produção de jogos eletrônicos e uma breve história dos jogos é abordada. O segundo capítulo do trabalho apresenta o desenvolvimento do jogo ***Os 5 Desprezíveis***. Por fim, o terceiro capítulo apresenta os resultados e conclusões desse projeto.

No capítulo 1, referente à fundamentação teórica, é apresentada de maneira resumida a história das jogos eletrônicos, buscando abordar personagens chaves para o crescimento da indústria, sem esquecer das grandes empresas que acrescentaram valor histórico para esse mercado. Alguns problemas do desenvolvimento de jogos são apresentados. São abordados temas como o *crunch time*, o ambiente de trabalho e a multidisciplinaridade. Ainda nesse capítulo são abordados algumas metodologias para resolver problemas como a comunicação e testes alpha e por último são apresentados o ciclo de vida do desenvolvimento de jogos e diversos processos para a produção de jogos. Os modelos *Cascata*, *Extreme Game Development*, *GUP* e *Game Scrum* são explorados.

No capítulo 2, referente ao desenvolvimento, a pré-produção é explorada, mostrando os principais acontecimentos que acarretaram em decisões futuras e na forma com que a produção foi conduzida. Os passos para a construção do jogo, ***Os 5 Desprezíveis***são documentados, expondo soluções encontradas para problemas encontrados ao longo da produção.

No capítulo 3, os resultados da produção do jogo são apresentados, abordando os testes, a arte, a jogabilidade e a utilização de ferramentas ao longo do desenvolvimento. Também é apresentado o ***Os 5 Desprezíveis***como um todo e as conclusões da produção.

1 Fundamentação Teórica

1.1 Breve história dos jogos eletrônicos

Segundo (KENT, 2001), o primeiro foi criado em 1962, e se chamava ***Spacewars***. Criado por Steve Russell dentro do MIT (Instituto de Tecnologia de Massachusetts), o jogo consistia em um combate de duas naves no espaço. O objetivo era simples: destruir o oponente antes que ele fizesse o mesmo com você. Fora o inimigo, ainda havia um sol, com gravidade que influenciava a nave e seus torpedos, Figura **??**.

Russell havia criado o ***Spacewars*** apenas como prova de suas habilidades como *hacker*, sem realmente cogitar a possibilidade de comercializar o produto criado. Russell se quer chegou a patentear a sua criação. Devido ao custo computacional na época, para rodar o jogo era necessário um PDP-1, o qual custava cerca de U$120,000.00. ***Spacewars*** nunca foi lançado como um game comercial, sendo utilizado apenas como um programa diagnóstico dos computadores PDP.

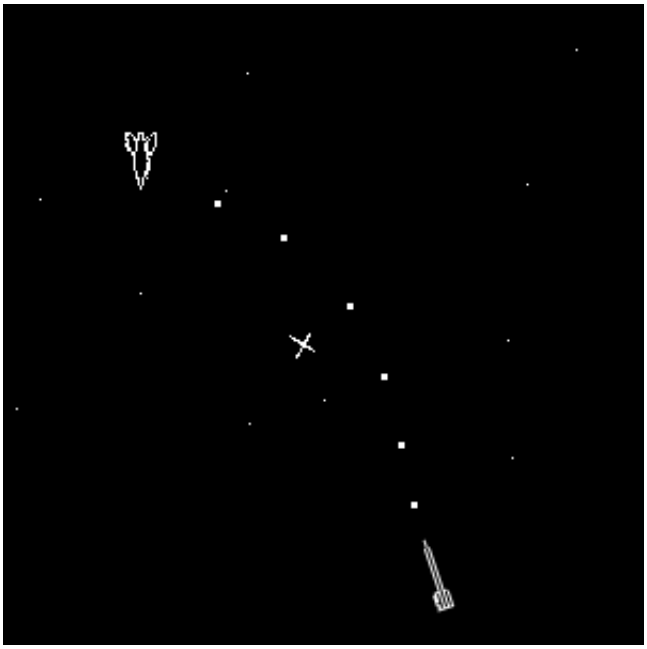

Figura 1 – Spacewars

O primeiro console de vídeo game foi criado por uma empresa de equipamentos militares, denominada Sanders (HAGIU; HALABURDA, 2009). O engenheiro Ralph Bear trabalhou durante 4 anos em um produto que pudesse criar jogos emitindo imagens para um televisor. O console foi comprado e distribuído pela empresa Magnavox, especializada em fabricar televisores. Ele foi batizado com o nome de Magnavox Odyssey e era vendido por U$100,00. A comercialização começou em 1972.

Devido a primitividade da ideia e a falta de conhecimentos de *game design*, os jogos sempre consistiam em dois retângulos brancos que podiam ser movidos livremente pela tela e em folhas semitransparentes. As folhas semitransparentes, que eram colocadas diretamente no televisor, formavam mapas, quadras, labirintos e outros locais onde o jogo deveria ocorrer. Devido a esse formato, as regras do jogo estavam apenas na cabeça dos jogadores, sendo possível trapacear a qualquer momento. Como um todo, o Odyssey nunca foi um grande sucesso comercial, embora tenha grande importância histórica.

Em 1971, Nolan Bushnell, um recém formado engenheiro eletricista, consegui construir uma máquina capaz de rodar um clone do jogo ***Spacewars***, chamado de ***Computer Space***, com um custo comercialmente viável. A máquina foi vendida por U$1500,00, e mesmo sendo uma grande inovação para a época, os controles complexos e a estranha aparência da máquina fizeram que o primeiro arcade[1] não fosse um grande sucesso(BAER; BURNHAM, 2001). Mesmo com a falha em sua primeira investida comercial, Nolan Bushnell criou uma empresa voltada apenas ao desenvolvimento de arcades, a Atari, em 1972, junto com mais dois amigos.

O primeiro jogo criado pela Atari, ***Pong***, foi um sucesso (KENT, 2001). Ao contrário de ***Computer Space***, o jogo era extremamente simples, sendo que a única instrução para jogá-lo era "*Avoid Missing Ball For High Score*". Até 1974, a Atari fabricou basicamente modificações do ***Pong***; os demais arcades fabricados não chegaram perto do sucesso de Pong, muito embora jogos como ***Trak 10*** e ***Gotcha*** tenham feito muito sucesso. Em 1975, a Atari lançou o Home Pong no natal, outro grande *hit*, Figura 2.

Figura 2 – Home Pong

[1] A palavra arcade é derivada da noção de uma casa com várias máquinas operadas por moedas. Como nessas casas as máquinas eram dispostas de forma a formarem colunas, ou arcadas, o nome arcade rapidamente se tornou sinônimo para essas máquinas.

Em 1977, a Atari lançou o Atari VCS, também conhecido como Atari 2600. Esse console teve um grande impacto no mercado, uma vez que o sucesso do Home Pong o precedia. Inúmeros portes de grandes jogos de arcade foram feitos para o VCS. Outro fator decisivo para o sucesso do VCS foi o licenciamento da plataforma. Empresas como a Eletronic Arts e a Actvision, essa última formada por ex-funcionários da Atari, foram responsáveis por diversos jogos de sucesso para o VCS.

O sucesso do VCS se manteve até 1982, quando o mercado norte americano de videogames simplesmente deixou de crescer. Durante o seu período de sucesso, a Atari se tornou a empresa com o crescimento mais rápido dos Estados Unidos (KENT, 2001), tendo um valor comercial de 2 bilhões de dólares. Várias polêmicas e grandes eventos circundaram a Atari, casos como a demissão do seu fundador, Nolan Bushnell; o evento Sword Quest, que premiaria os jogadores com até U$150.000,00 em um torneio, e o lançamento do jogo ***E.T***, considerado como o pior jogo da história, são apenas alguns dos fatos que envolveram a Atari durante esse período.

A época de 1978 até 1983 é também conhecida como a grande era dos videogames, ou a era de ouro. Isso se dá devido à explosão de jogos lançados e enorme interesse pelos videogames. Inúmeras casas de arcades, ou fliperamas, foram criadas e inúmeros consoles foram lançados. Boa parte desse sucesso se deve a alguns games que levantaram a indústria. Para citar alguns:

- ***Space Invaders***: Lançado pela Taito, uma empresa russa, foi uma grande sucesso no Japão e no mundo. No Japão causou uma escassez de moedas e fez com que muitas lojas pequenas deixassem de vender vegetais e outros produtos para instalar máquinas de Space Invaders. Cerca de 1.000.00 unidades deste arcade foram vendidas.

- ***Pac Man***: Lançado pela Namco, o jogo foi feito com a intenção de abraçar o mercado de jogadoras. Contendo apenas o tema de comer, o jogo tinha um caráter não violento. Contendo uma jogabilidade muito simples, o jogo consistia em comer 240 pontos dentro de um labirinto sem ser capturado pelos fantasmas. Vendeu mais de 100.000 unidades apenas nos Estados Unidos, sendo que a continuação, Ms. Pac Man, manteve o sucesso, vendendo 115.000 unidades no Estados Unidos. Pac Man ainda é considerado por muitos o símbolo da indústria dos games.

- ***Donkey Kong***: Lançado pela Nintendo, o jogo consistia em um carpinteiro (***Jumpman***) que estava atrás de sua namorada, que havia sido sequestrada por um gorila, Donkey Kong. O jogo enfrentou um processo contra a Universal, que supostamente possui os direitos do King Kong. A Nintendo ganhou a causa e manteve um dos seus maiores símbolos. O jogo de arcade vendeu 75.000 unidades em arcades.

Em 1983 o mercado norte americano parou de crescer. Segundo (III, 1989) os motivos para a queda do mercado podem ser classificados como:

1. Gráficos fracos e jogabilidade superficial.
2. Os jogos eram melhores nos arcades.
3. Poucas cores.
4. Qualidade de som baixa, efeitos sonoros limitados.

O mercado só voltou a crescer nos Estados Unidos com o NES (Nintendo Entertainment System). O NES se tornou o vídeo game mais vendido até o momento: 1,8 milhões de consoles em 1986; 5,4 milhões em 1987; 9,3 milhões em 1988, e em 1990 o total de vendas chegou a 13,9 milhões de consoles apenas nos Estados Unidos. No final da produção do NES o total de vendas chegou a 60 milhões de unidades em todo mundo (NINTENDO, 2013).

Até o ano de 1990, a Nintendo se manteve como a maior líder de mercado, tendo apenas a Sega, com o console Master System, como concorrente. A partir de 1990 até 1994, a hegemonia da Nintendo teve um fim, com a Sega obtendo cerca de 55% do mercado norte-americano. Isso foi possível devido ao novo console Genesis, e a três fatores que apoiaram a campanha de lançamento do sistema: a forte propaganda da experiência similar aos arcades, caracterizada pelo slogan "*Genesis does what Nintendon't*" (HAGIU; HALABURDA, 2009); a presença de grandes artistas nos jogos, como Michael Jackson em ***Moonwalker***; e a presença de jogos de alta popularidade na compra do console, como o jogo ***Sonic the Hedgehog***.

Em 1994, a Sony, grande empresa de eletrodomésticos, entrou na briga de consoles com o Playstation, produto de uma parceria não concretizada com a Nintedo. Em 2001, devido ao estado financeiro da empresa e a derrota de vendas de seu último console, o Dreamcast, em relação ao seu principal concorrente, o Plastation 2, a Sega deixou de fabricar *hardware* e passa a produzir *software* para as demais empresas (KENT, 2001).

A partir da entrada do Playstation, em 1994, até o lançamento dos consoles conhecidos como sétima geração de consoles, a Sony se manteve estável com a maior parte do mercado de games. Sempre adotando a política de licenciamento U$10,00 (a cada jogo vendido para o Playstation, a Sony ganha 10 dólares da transação), a Sony Games se tornou um mercado viável para novas empresas produzirem conteúdo de maneira lucrativa.

A sétima geração de consoles marca o retorno da Nintendo como a líder no mercado de videogames. Com 32 milhões de unidades vendidas do console Wii até o ano de 2012, a Nintendo estava muito à frente das suas principais concorrentes: o Playstation 3 da Sony

com 15,5 milhões e o Xbox 360 da Microsoft com 20,9 milhões (HAGIU; HALABURDA, 2009).

Essa geração possuiu um grande diferencial no modo de controle do usuário com seus jogos. O Wii possui um sensor de movimento em seu controle, podendo simular diversos tipos de movimento com o braço. Já o Xbox 360, através do Kinect, é capaz de reconhecer uma pessoa por inteiro e utilizar todo o corpo para controlar o jogo em questão. Na data de publicação desse trabalho, a geração atual é a oitava geração, tendo como principais consoles o Plastation 4, o Xbox One e o WiiU.

1.2 Problemas do desenvolvimento de jogos

O desenvolvimento de jogos deve seguir metodologias de Engenharia de Software para garantir uma maior qualidade final, uma vez que o jogo eletrônico em si é um *software*. No entanto, um jogo eletrônico possui algumas diferenças do desenvolvimento tradicional de software fazendo com que metodologias específicas sejam usados em projetos e de jogos eletrônicos.

Segundo (GODOY ANDRé, 2010), existem basicamente 3 grandes problemas comuns no desenvolvimento de jogos. O primeiro deles está relacionado ao escopo do projeto, o segundo ao planejamento e o terceiro ao *crunch time*. De certa forma podemos endereçar todos esses problemas como decorrência do erro de planejamento, uma vez que é nesse momento o escopo pode ser alterado para o ideal e assim evitar o *crunch time*, período de grande carga de trabalho. É necessário notar que o planejamento pode ocorrer de forma iterativa, o que diminui o risco de ocorrer nesses problemas.

No entanto, outro problemas podem ser levantados ainda no desenvolvimento de *software* na área de mídia, acarretando problemas para o desenvolvimento de jogos eletrônicos. Os problemas podem ser relacionados a presença de artistas no time, peça fundamental para qualquer trabalho de entretenimento, e a grande dúvida dos requisitos e da expectativa do usuário do sistema (BISWAS; SINGH, 2006). Esses problemas, comumente, são resolvidos através da prototipação evolutiva. Porém, esse tipo de abordagem pode trazer grande custos devido a interpretação errônea de requisitos entre engenheiros e artistas, alto custo de recriação de protótipos funcionais e o alto custo na validação dos protótipos (BISWAS; SINGH, 2006).

Dessa forma o processo utilizado para o desenvolvimento de jogos deve primar em resolver os seguintes itens: comunicação entre um time multidisciplinar (artistas, engenheiros, produtores e outros); escolher a melhor *feature* em tempo hábil e exequível. Questões relevantes ao escopo também devem ser consideradas mesmo durante o desenvolvimento de cada *feature*: a importância da *featuer* perante o jogo final, revisão da mesma mediante a alteração artística.

1.3 *GameFlow*

A descrição de diversão em modelos utilizada dentro da área de jogos é um estudo, por muitas vezes, disperso. Existem algumas formas de se classificar diversão em outras áreas de conhecimento, como os modelos de teoria disposição, atitude, a teoria de transporte, cognição, interação parassocial e fluxo (OLIVER, 2004).

O *GameFlow* é um modelo para definir a diversão dentro do contexto de jogos. Derivado do modelo de fluxo, o *GameFlow* se caracteriza por definir 7 elementos básicos que geram diversão (SWEETSER, 2005). Segundo o modelo, para que determinado jogo seja divertido, é preciso que cada elemento esteja presente e tenha interação com um ou mais elementos. Os elementos do modelo estão apresentados na Tabela 1.

Elementos do modelo	**Descrição**
O jogo	Uma tarefa que pode ser concluída
Concentração	Capacidade de se concentrar na tarefa
Desafio	Habilidades percebidas devem corresponder aos desafios e ambos devem exceder um determinado limiar.
Controle	Permite exercer um senso de controle sobre as ações
Metas claras	A tarefa deve ter metas claras
Feedback	A tarefa fornece um *feedback* imediato
Imersão	Envolvimento profundo mas sem esforço, preocupação e sentido de tempo reduzidos
Interação Social	-

Tabela 1 – Elementos do *GameFlow*

Com esse conjunto de elementos é possível criar uma forma de avaliação de determinado jogo para se definir se esse jogo é divertido ou não, segundo o modelo. Essa abordagem tem o intuito de mitigar a prototipação exacerbada. Para tanto, o *GameFlow* possui os critérios apresentados na Tabela 2.

1.4 *Game Design Document*

O desenvolvimento de um jogo é pautado em um documento que sumariza todo o projeto, conhecido como *Game Design Document*, GDD. Esse documento serve como central de documentação para toda a equipe, descrevendo com clareza todos os itens presentes no jogo (BALDWIN; CONSULTING, 2005).

Atualmente, o GDD é mais visto como forma de documentação do que uma forma de comunicação (MANKER, 2011), sendo que esse trabalho fica para outras ferramentas,

Elementos do modelo	**Critérios**
Concentração	Os jogos devem: • Fornecer uma série de estímulos de diferentes fontes • Fornecer estímulos que valem a pena lutar • Pegar rapidamente a atenção dos jogadores e manter o foco durante todo o jogo • Ter uma alta carga de trabalho, embora ainda seja apropriado para percepção dos jogadores Os jogadores: • Não devem ser distraídos de tarefas que eles querem ou precisam se concentrar • Não devem ser sobrecarregados com tarefas que não são importantes
Desafio	• Desafios em jogos devem coincidir com os níveis de habilidade dos jogadores • Jogos devem fornecer diferentes níveis de desafio para jogadores diferentes • O nível de desafio deve aumentar à medida que o jogador progride através do jogo e aumenta seu nível de habilidade • Jogos devem proporcionar novos desafios em um apropriado passo
-	• Jogadores devem ser capazes de começar a jogar o jogo sem ler o manual • Aprender o jogo não deve ser chato, mas fazer parte da diversão • Jogos deve incluir ajuda para que os jogadores não precisam de sair do jogo • Os jogadores devem ser ensinados a jogar o jogo através de tutoriais ou níveis iniciais • Jogos deve aumentar as habilidades dos jogadores em um ritmo apropriado, à medida que eles progridem através do jogo • Os jogadores devem ser recompensados de maneira adequada por seu esforço e pelo seu desenvolvimento de habilidades • Interfaces de jogo e mecânica deve ser fácil de aprender e usar

como o desenho conceitual, *story boards* e outros.

Existem inúmeras formas de se escrever um GDD, no entanto alguns itens devem ser abordados em sua criação. Dentre os vários pontos que o GDD deve compreender estão citar (ROGERS, 2010):

- O nome do jogo, mesmo que não seja o nome de lançamento do jogo.
- Objetivos do jogo, a principal meta do jogador para com o jogo.
- Uma visão geral da história, mostrando claramente o início e o fim do jogo, caso existam.
- Os controles utilizados pelo jogo.
- O formato da câmera e seu posicionamento, fotografia.
- Requisitos de tecnologia, os quais também devem estar presentes em um documento específico para tecnologia, o documento de design técnico (TDD).
- Fluxograma de cenas, mostrando quais são as possibilidades de navegação entre diversas cenas do jogo.
- A descrição dos personagens do jogo, principalmente do personagem que o jogador irá interpretar e incluindo o maior número possível de NPCs.
- Métricas do jogador, como a sua velocidade e o seu tamanho relativo ao mapa.
- *Power-Ups*, itens capazes de tornar o jogador mais capacitado para determinado cenário do jogo.
- Mecânica universal, formato seguido pelo jogo, caracterizado em tipos de jogo, como tiro em primeira pessoa, estratégia em tempo real, etc.
- Os níveis que o jogador percorre.
- Segredos.
- Forma de pontuação.
- Lista de todos inimigos e sua descrição.
- Músicas e efeitos sonoros.

1.5 Ciclo de vida de jogos

Ciclos de vida são formas de descrever a vida de um produto, desde sua concepção até o momento em que ele chega ao mercado consumidor ou quando ele deixa de ter utilidade. Muito próximo ao ciclo de vida de um simples produto de engenharia, o jogo eletrônico costuma seguir o ciclo de vida apresentado abaixo:

1.5.1 Pré-Produção

A pré produção é o momento em que acontece a definição das principais *features*, do apelo comercial e da identidade do jogo. O chamado *fun factor* (GODOY ANDRé, 2010) deve ser encontrado o mais cedo possível, de preferência durante essa fase do desenvolvimento.

O mais importante documento do desenvolvimento, documento de *design* do jogo, deve ter sua primeira versão criada. Nessa fase, também são criados documentos importantes, como o cronograma, Documento de *Design* Técnico, e os demais documentos que irão auxiliar na produção.

Aqui as principais decisões comerciais são feitas, como o valor de marketing, o valor máximo a ser alcançado com o desenvolvimento do jogo dentre outras (BETHKE, 2003).

1.5.2 Produção

Esse é a fase onde o GDD é convertido em *software*. Aqui ocorre a codificação, a produção artística, testes. Essa fase representa boa parte do desenvolvimento em si, e por isso ela deve seguir um processo capaz de prever e lidar com as eventuais falhas de desenvolvimento. Nesse ciclo de vida, qualquer metodologia ou processo pode ser aplicado para essa fase. Exemplos são o Cascata (FLOOD, 2013) e o eXtreme Game Development (DEMACHY, 2013).

Nesse momento os problemas de planejamento, as complicações na comunicação entre a equipe e os *crunch times* irão parecer. Portanto é necessário combinar metodologias para enfrentar esse tipo de acontecimento ou mesmo previní-los, se possível. Um exemplo de medida para evitar as possíveis falhas de comunicação entre os engenheiros e os artistas é a rápida prototipação de baixa fidelidade (BISWAS; SINGH, 2006), que consiste basicamente de um protótipo feito para uma *feature* ou um item muito específico que gera alguma discordância dentro do time. Os testes alfa e beta também são realizados durante a produção.

1.5.3 Pós-Produção

O objetivo da pós-produção é a venda do jogo e a sua manutenção. Dessa forma, as área de marketing e logística são empregadas de maneira mais extensiva.

Outra atividade importante é a criação do *post-mortem*. O *post-mortem* é um documento que descreve as melhores e as piores partes do desenvolvimento, bem como as lições aprendidas. Também é muito comum a criação do *post-mortem* de maneira evolutiva, para que no final apenas seja concretizado o conhecimento do projeto.

A manutenção pode acontecer de diversas formas. Um exemplo é o suporte dado aos jogos de hoje através do uso de *patches* para a correção de *bugs*. Em jogos de MMORPG é comum o lançamento de *updates* mensais para equilibrar os diversos componentes do jogo.

1.6 Processos de desenvolvimento de jogos

Um processo de Engenharia de Software normalmente remete a um modelo. Modelos são implementações de determinado ciclo de vida em um contexto especifico. Exemplo de processo comuns em Engenharia de Software são: Processo Unificado (PU) (JACOBSON; BOOCH; RUMBAUGH, 1999), Processo Cascata (LAPLANTE; NEILL, 2004) e Rapid Application Development (RAD) (TANG, 1997).

Atualmente existe uma espécie de cisão entre dois tipos de modelos de processos: processos tradicionais e as metodologias ágeis. Os processos tradicionais são aqueles inspirados, em grande parte, nos modelos tradicionais de desenvolvimento de produtos de engenharia, como a linha de montagem fordista. Esses modelos são conhecidos por ter uma documentação forte, por uma separação de papéis clara e por ter a mudança como um grande risco.

As metodologias ágeis são conjuntos de atividades que procuram seguir 4 princípios, relatados no manifesto ágil (BECK, 2013). Essas metodologias tendem a aceitar de maneira mais simples uma mudança, uma vez que seus processos visam sempre a presença de uma grande quantidade de mudanças durante o desenvolvimento.

Devido a especificidade do desenvolvimento de um jogo eletrônico, existem alguns poucos modelos de processo para essa área. Exemplos desses casos são: *Cascata*, *Extreme Game Development* (DEMACHY, 2013), *Game Unified Process* (FLOOD, 2013), *Game Scrum* (GODOY ANDRé, 2010) e*Agile Game Process* (SANTOS, 2006).

1.6.1 Cascata

O modelo cascata é o modelo tradicional do mercado (FLOOD, 2013). Tendo um sistema direto de transição, a sua assimilação é simples. No entanto, devido a linearidade inerente as suas fases, mudanças e imprevisibilidade no projeto podem causar o seu fracasso.

Este modelo pode ser caracterizado pelas seguintes atividades, que ocorrem na sequência apresentada abaixo(FLOOD, 2013):

1. **Concepção**: o gerente, a equipe e os *stakeholders* se reúnem para discutir como o jogo de ser. Itens discutidos: público alvo, plataforma, tempo de desenvolvimento, aspectos técnicos e artísticos de alto nível, e algumas *features*.

2. **Especificação do Jogo**: criação do Documento de Design do Game.

3. **Bíblia de arte e história**: aqui a identidade visual, as ferramentas, e tudo o que estiver relacionado diretamente a arte visual e sonora é documentado. O mesmo é feito com a história do jogo.

4. **Especificação Técnica**: aqui a documentação irá expressar a arquitetura do jogo em diagramas UML ou em outra forma de diagramas. O objetivo é ter a arquitetura definida antes da construção.

5. **Construção**: aqui acontece o desenvolvimento de tudo o que foi definido através da especificação do jogo, das bíblias e da especificação técnica.

6. **Controle de Qualidade**: o controle de qualidade verifica se a documentação está coerente com o jogo no estado atual. Caso não esteja, o responsável é avisado do fato. Fora essa verificação, outros métodos são utilizados para validar o jogo, são eles:

    a) **Teste de jogo**: Um grupo pequeno de especialistas é chamado para avaliar o estado do jogo e verificar a possibilidade de presença de *bugs* ou *gliches*.

    b) **Teste Alfa**: Um grupo reduzido de usuários finais é escolhido para reportar a sensação de jogar o jogo. Também funciona como um teste de jogo.

    c) **Teste Beta**: O jogo é liberado, porém o estado beta mostra que os aprimoramentos finais não foram feitos. Dessa forma a *release* pode conter *features* ou artes mais aprimoradas.

7. ***Release***: versão final do game.

1.6.2 *Extreme Game Development* (XGD)

O XGD é derivado da metodologia ágil XP. O XP (eXtreme Programming) foi criado por Kent Back enquanto trabalhava na Crysler (BECK; ANDRES, 2004). Dentre os objetivos dessa nova metodologia estavam: conciliar humanidade com produtividade, mudança social e melhoria. Para tanto, essa nova forma de desenvolvimento foi criada com base em 5 valores: comunicação, simplicidade, *feedback*, coragem e respeito. Outra característica importante do XP é o fato que o *software* pertence a toda equipe: o código, o sucesso e o fracasso são compartilhados por todos. Da mesma forma, todos da equipe devem codificar, testar, ouvir e projetar continuamente.

Muito embora o XP descreva como decorre o trabalho diário, pouco é falado de como a gerência do projeto funciona em um projeto que usa essa metodologia. Essa área fica a cargo da empresa escolher a metodologia mais adequada para sua realidade.

O XGD foca principalmente em adaptar técnicas XP para o contexto de *game design* e criação de conteúdo multimídia e para o uso de testes automatizados para características do jogo, como o *fun factor*. Em especial, o XGD traduz os valores do XP para o contexto de jogos (DEMACHY, 2013).

1. **Simplicidade**: o código criado deve ser simples, principalmente se ele irá ser usado apenas uma vez. Ao contrário da ideia de se programar para interfaces, e criar objetos totalmente genéricos e adaptáveis, é necessário pensar antes se esses objetos serão utilizados mais de uma vez. Outro exemplo é que se um prédio faz parte do *background*, o seu interior não precisa ser modelado.

2. **Comunicação**: fato crucial devido a multidisciplinaridade da equipe. Documentos devem ser mantidos, mas todos devem ser curtos. É mais importante conversar sobre uma ideia do que catalogá-la. Se um documento é criado, seja o documento de *design* técnico ou mesmo o GDD, ele deve ter seu tamanho reduzido para que qualquer um possa ler ele.

3. ***Feedback***: a resposta para o processo de desenvolvimento deve estar clara não apenas na pós-produção, mas ao longo de toda produção. Iterações de avaliação do desenvolvimento devem ser realizadas buscando encontrar 5 pontos positivos que devem continuar sendo executados e 5 pontos negativos que devem ser avaliados.

4. **Coragem**: coragem dos *stakeholders* para entender que as metas não são permanentes. E por fim, coragem para os desenvolvedores aceitarem *feedback* regular e autocrítica.

1.6.3 *Game Unified Process* (GUP)

O GUP não é nada mais que o esforço de se utilizar o processo unificado (UP) com algumas características do XP para o desenvolvimento do jogos. O seu conceito é apresentado em (FLOOD, 2013).

O processo unificado foi a tentativa de unir diversas correntes de modelos de processo do final da década de 90, tendo a sua primeira descrição formal no livro (JACOBSON; BOOCH; RUMBAUGH, 1999). Devido ao seu objetivo, o UP tem uma enorme diversidade de papéis, artefatos, tarefa e atividades. Com isso, muitos projetos falham em usar o UP quando tentam usar todas as suas características no mesmo projeto.

Em uma visão geral, o UP pode ser dividido em fases, iterações e disciplinas. Suas fases são: iniciação, elaboração, construção e transição. O GUP implementa a pré-produção nas fases de iniciação e elaboração. Essas fases são focadas na construção de um modelo de negócio viável e a criação de uma concepção válida para o jogo. A produção ocorre durante a construção e em parte da transição. O que é feito durante as fases é descrito nas disciplinas. Iterações são períodos definidos dentro de uma fase onde determinadas atividades de algumas disciplinas são executadas.

As disciplinas do UP são: modelagem de negócio, requisitos, análise e *design*, implementação, teste e implantação. Em cada uma das disciplinas, uma parte de uma fase do modelo cascata é executado, dessa forma em uma iteração diversas atividades relevantes são executadas. O relacionamento entre as fases do modelo cascata e as disciplinas no GUP podem ser descritas na Tabela 3:

Cascata	**GUP**
Concepção	Modelagem de Negócio, Requisitos
Especificação do Jogo	Análise e *design*, modelagem de negócio, requisitos
Bíblia de arte e história	Requisitos, implementação
Construção	Análise e *design*, implementação, testes
Controle de Qualidade	Implementação, testes
Release	Implementação, implantação

Tabela 3 – Relacionamento de fases GUP com as fases do Cascata

A Figura 3 mostra a interação entre as fases e disciplinas em relação ao tempo no UP.

1.6.4 *Game Scrum*

O *Game Scrum* é uma adaptação do ciclo de vida do desenvolvimento clássico de jogos para um contexto ágil. Como o nome sugere, a metodologia ágil por trás do *Game Scrum* é o próprio *Scrum*.

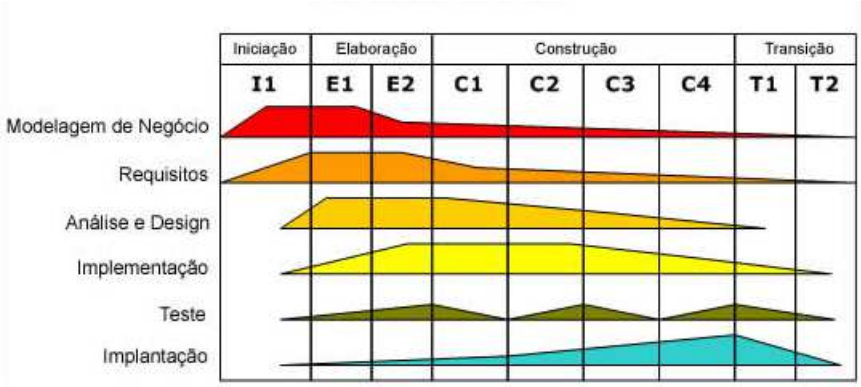

Figura 3 – Interação entre as fases e disciplinas em relação ao tempo no UP

O *Scrum* é uma metodologia de gerenciamento, não necessariamente de *software*, em que a presença de constantes de mudanças de requisitos está presente. Constituído por apenas 3 papéis, 3 artefatos e 5 cerimônias, o *Scrum* se destaca por ser extremamente simples e adaptável.

Os papéis do *Scrum* são *Product Owner*, *Scrum Master* e o Time. O time é o conjunto de todos os responsáveis diretos por se produzir o *software*, desenvolvedores, gerentes, analista de requisitos e outros. Como só existe um papel para todas essas pessoas, a classificação de desenvolvedor, analista e outros não é necessária e deve ser evitada. O *Scrum Master* faz parte do time, mas também tem o dever de fazer a comunicação entre o *Product Owner*, coordenar as cerimônias do *Scrum* e manter o compromisso da equipe com o *Scrum* e com o projeto. O *Product Owner* é o principal interessado com o sucesso do *software*. Representa todos os *Stakeholders* faz parte da equipe e deve estar comprometido e inteirado com o processo.

Os artefatos do *Scrum* são: *product backlog*, *sprint backlog* e *burndown*. O *product backlog*, também chamado apenas de *backlog*, é uma lista de requisitos do *software* expresso da maneira mais simples para o *Product Owner*. O *sprint backlog* é o conjunto de itens retirado do *backlog* que irá ser implementado na próxima *sprint* . O *burndown* é um gráfico que representa quanto trabalho falta para alcançar o objetivo da *sprint*, o *burndown* ainda pode representar a produtividade da equipe, ser usado para prever possíveis atrasos e ainda representar o projeto em *sprints*.

As cerimônias do *Scrum* são: *sprint*, *sprint planning*, *sprint review*, *daily meeting*, *sprint* retrospective. A *sprint* é um tempo período de duração fixa onde o desenvolvimento ocorre. Costuma ter de 1 a 4 semanas de duração, sendo a duração definida na *sprint planning*. *A sprint planning* é o momento de definir o tamanho, em tempo e em esforço que será empregado para realizar os principais itens escolhidos pelo *Product Owner*. O *sprint review* é o momento que o time se reúne para verificar o que foi feito e o que não foi feito, bem como para apresentar os resultados para os interessados. O *daily meeting* é

um momento em que o time se reúne por 10 minutos para assumir o compromisso sobre o que será realizado durante o dia e informar o que foi feito e os possíveis obstáculos encontrados durante o dia anterior. Por fim, a retrospectiva da *sprint* é o momento em que a equipe se reúne para rever a *sprint* e entender os erros e fortalecer aquilo que foi aprendido.

A Figura 4 resume a estrutura do *Scrum*.

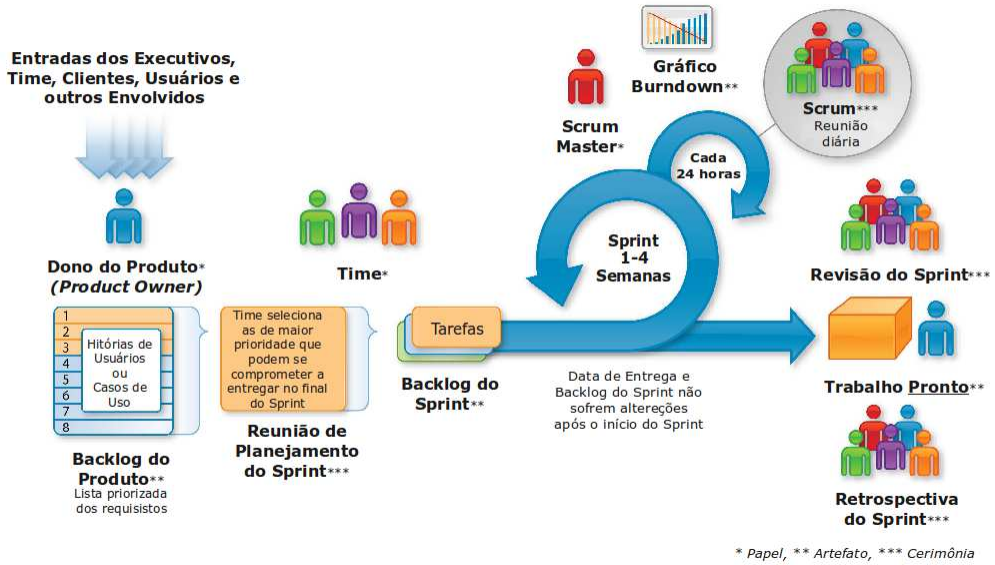

Figura 4 – Estrutura do *Scrum*

O *Game Scrum* tem como forte o fato de seu valor incremental ser valido como forma de prototipação. Outros valor importantes como o uso do *Kanbam* para serializar e controlar o trabalho artístico (KEITH, 2013), a construção gradativa do *post-mortem* através das *sprints reviews*, podem ser facilmente implantados nessa metodologia.

2 Desenvolvimento

2.1 Metodologia

Para a realização do projeto, que iria durar cerca de 3 meses no período de produção, alguns quesitos foram previamente definidos. O processo de desenvolvimento deveria ser escolhido conforme a equipe, com o intuito de mitigar o risco da não aceitação da forma de trabalho.

Contudo, certos práticas de metodologias ágeis seriam utilizados independente do processo escolhido. A produção e entrega de executáveis em curtos períodos, prototipação rápida e momentos de refatoração são exemplos de praticas realizadas durante todo o período de produção dentro do desenvolvimento do ***Os 5 Desprezíveis***.

Durante a pré-produção, o trabalho previa uma documentação mais criteriosa dos trabalhos realizados ao longo da produção, bem como dos requisitos e *features* do jogo. No entanto, a forma de documentação seria decidida com base no processo escolhido. Apenas o documento de *design* do jogo tinha sido construído e mantido pela equipe. Uma vez que não existe um modelo fixo, apenas guias com conteúdo aconselhado, esse documento foi construído procurando ser o mais sucinto possível, evitando descrições detalhas de *level design*, sonografia por exemplo.

Na produção, com o processo a ser utilizado já definido, Scrum, foi inserido uma forma de avaliar o desempenho para certificar o quanto seria possível realizar do GDD proposto. Embora não houvesse histórias de usuário[1] para cada *feature*, existiam pontos dentro do GDD que indicavam claramente a completude do jogo e que orientavam, de maneira priorizada, o que deveria ser construído. Esses itens eram, principalmente, os capítulos de investigação e de acusação. Dado esses itens, foi possível avaliar, mesmo que de maneira muito primitiva, a evolução do código do capítulo de investigação e assim criar um comparativo de desempenho para o de acusação. Esses capítulos serão discutidos na seção de produção deste trabalho.

Na pós-produção foram medições de qualidade do jogo, uma vez que nesta fase foram realizados os testes alfa. Para realizar tal tarefa, um questionário foi criado com base na metodologia *GameFlow* apresentada na fundamentação teórica.

[1] História de usuário são algumas frases que capturam uma necessidade do usuário no seu linguajar cotidiano

2.1.1 Ambiente

Para realização desse projeto foram utilizados uma serie de ferramentas. Para a manipulação de imagens foi utilizado o Gimp[2] versão 2.8.6, utilizando o filtro de ruído HRV e dessaturação. Para os efeitos sonoros, foi utilizado o Audacity[3] para cortes e efeitos de *fade in* e *fade out*. Para construção do jogo e prototipação foi utilizada a *engine* Unity3D[4], versão 4.2. Para edição de scripts foi utilizado o Sublime Text 3. O versionamento do trabalho, principalmente do código fonte, foi feito através do GIT[5]. Para acompanhar a evolução do ***Os 5 Desprezíveis***e para definir tarefas foi utilizado o Scrummy[6] e o Yodiz[7].

2.2 Pré-produção

Uma vez iniciado o projeto, o passo seguinte foi discretizar o projeto nos moldes apresentados por (RABIN, 2010), com exceção da fase de concepção, que foi incorporada a pré-produção. Dessa forma três períodos distintos estiveram claramente presentes: pré-produção, produção e pós-produção.

2.2.1 Equipe

Para a construção do ***Os 5 Desprezíveis***, inicialmente foi escolhida a equipe que iria trabalhar no desenvolvimento. Com o intuito de minimizar lacunas de projeto e de construir um conceito condizente com a capacidade da equipe, os principais aspectos para seleção foram a disponibilidade e a capacidade artística.

A equipe inicialmente era constituída por dois programadores, um artista e um redator. Ciente das limitações da equipe, principalmente as ligadas ao limite artístico (devido a quantidade de artistas presentes no projeto), optou-se por desenvolver um jogo mais baseado em escolhas do que em belas apresentações de cenários. Dessa forma, o conceito inicial contemplava uma grande gama de possibilidades, *assets* e principalmente de mecânicas diversificadas para a jogabilidade. Parte desse conteúdo foi adicionada em um documento que se consolidou como a primeira versão do GDD.

Enquanto o desenvolvimento não tinha início, a equipe se baseou em focar nas ideias de *game design* e tornar o jogo o mais agradável possível no conceito, aprimorando o GDD. O início do desenvolvimento foi postergado durante 6 meses devido a parte da equipe estar envolvida em outros trabalhos. Essa abordagem gerou dois graves problemas:

[2] http://www.gimp.org/
[3] http://audacity.sourceforge.net/
[4] http://unity3d.com/
[5] http://git-scm.com/
[6] http://www.scrumme.com.br/
[7] http://www.yodiz.com/

1. Complexidade aberta de ideias

   Devido ao tempo gasto no aperfeiçoamento da ideia, a complexidade do conceito crescia de maneira incontrolada. Isso gerou um mal estar quando olhou-se o tamanho do investimento de tempo necessário para realizar o projeto. Como se acreditava que o jogo teria apenas valor comercial caso o os itens do GDD fossem seguidos, ficou claro a possibilidade de uma falha caso o jogo fosse encarado como uma investida comercial.

2. Compromisso entre a equipe

   Como a equipe não trabalhava diretamente com o desenvolvimento do ***Os 5 Desprezíveis***, tratando apenas do conceito, um sentimento de protelamento nasceu entre os membros do time. Isso levou ao distanciamento dos membros e a diminuição do tempo dedicado ao desenvolvimento do jogo por toda equipe.

Devido aos itens citados acima, a equipe inicial acabou por se desvencilhar antes do começo do desenvolvimento, restando apenas um redator e um programador. Com isso, o escopo foi revisto, o método de trabalho sofreu alterações e conceito do jogo foi reimaginado para viabilizar a sua realização. Isso acabou por mudar muito da mecânica do jogo, a forma com que eram adquiridas os *assets* de imagem e a forma com que a identidade do ***Os 5 Desprezíveis*** foi apresentada.

2.2.2 Forma de trabalho

Antes da dissolução da equipe, o método utilizado para gerir o trabalho ainda não estava definido. Isso ocorreu principalmente devido a relutância de parte da equipe em realizar todos os passos da metodologia inicialmente apresentada (Scrum). Os principais empecilhos foram a quantidade de reuniões e a necessidade de se produzir algo no período de uma *sprint*. Sem utilizar as reuniões diárias, a equipe se reunia através de vídeo conferência uma vez por semana.

Com o tamanho da equipe reduzido para um programador e um redator, iterações longas representavam um grande risco para a entrega de uma simples *build*. Os requisitos seriam repensados e seria necessário cortar muitos deles, já presentes no documento de *design* inicial. Com o novo cenário e considerando a experiência do programador com esta metodologia, a aplicação do *Scrum* se tornou uma opção viável.

As *sprints* foram reduzidas para períodos de uma semana, tendo as entregas como ponto de controle para o orientador desse trabalho. Isso forçou uma cultura de construir *builds* sempre executáveis e de prover um desenvolvimento constante de código e de conteúdo artístico.

No entanto, alguns conceitos básicos do *Scrum*, como o *Backlog*, não foram seguidos à risca. Isso aconteceu devido a quantidade resumida de integrantes e principalmente devido à incerteza no modo de separar corretamente e de maneira clara cada item dos passos do desenvolvimento, no contexto de jogos, em forma de história.

2.2.3 Primeiro GDD

A construção do primeiro documento de *design* do jogo foi feito com a equipe inicial. Por isso, muitas ideias foram desenvolvidas com uma proporcionalidade que não foi alcançada ao longo da produção. O GDD foi construído com o intuito de ser pequeno para evitar o caso "*write only, never read*" (NEVELSTEEN; GAYOSO, 2011).

2.2.3.1 Proposta Inicial

O jogo seria construído no estilo detetive para plataformas móveis, focando o jogo entre amigos e um alto nível de *replay*. As partidas seriam baseadas em histórias geradas aleatoriamente, fazendo com que cada partida fosse uma partida completamente diferente para cada jogador. Os jogadores interpretariam detetives que investigariam uma série de assassinatos, sendo que cada um dos personagens ao longo do jogo poderia ser o assassino, inclusive qualquer um dos jogadores.

2.2.3.2 Jogabilidade

O sistema do jogo seria baseado em decisões e procura por pistas através de cenários pré-definidos, bem como a implantação de falsas pistas para confundir os adversários. Como o jogador poderia ser um detetive e ao mesmo tempo ele poderia ser um assassino, o sistema se dividiria em caça e fuga.

Para a caçada, a cada ciclo de eventos, os jogadores estariam fazendo escolhas que iriam aproximar o detetive da resposta de quem é o assassino: o sistema ficaria responsável em disponibilizar as pistas de forma aleatória, fazendo com que existisse a necessidade de interação dos detetives para compartilhar pistas para se chegar ao assassino.

Para o sistema de fuga, o assassino deveria continuar a sua sequência de assassinatos até determinado número de ciclos de eventos ou até alguns dos detetives colocasse um inocente no seu lugar. Ele seria capaz de perseguir algum dos personagens do jogo e assassiná-lo, podendo até tirar um dos personagens, inclusive um dos detetives, da partida. Caso o assassino fosse um detetive, ele poderia plantar falsas pistas no cenário de crime para incriminar outros jogadores.

2.2.3.3 Apresentação visual

Para aumentar a imersão do jogo, toda a apresentação seria pautada no estilo gráfico *noir*, estilo de referência para HQs (revistas em quadrinhos) do gênero de investigação. Como não existia nenhuma necessidade de apresentar os jogadores com grande quantidade de movimento, os eventos iriam ocorrer como a leitura de uma HQ, forçando a leitura e escolhas inteligentes ao invés do uso de reflexos. Um exemplo do estilo é a apresentação do filme *Sin City* (Figura 5) ou do jogo *Max Payne* (Figura 6).

Figura 5 – Sin City

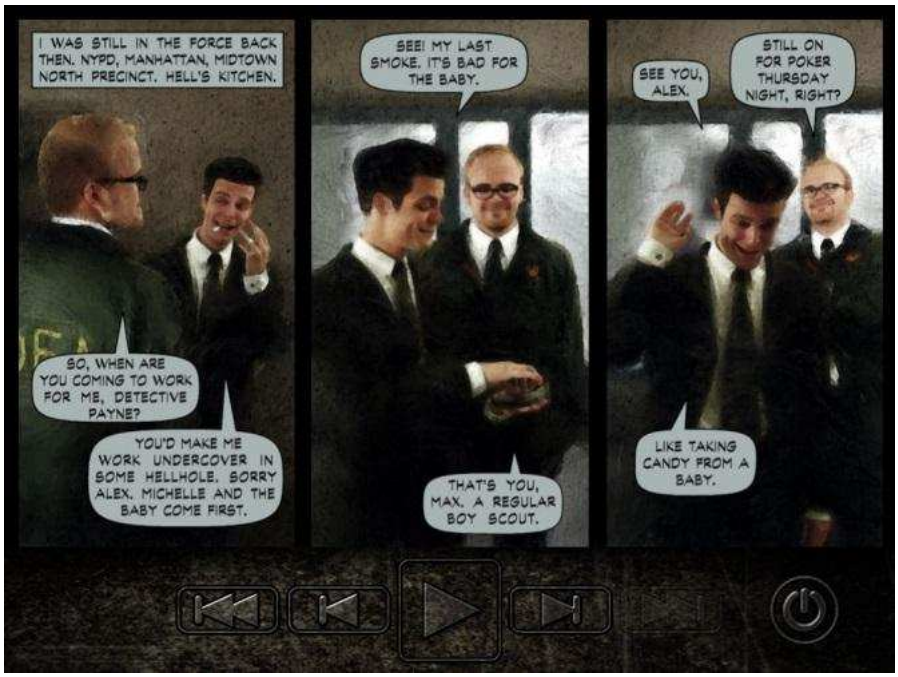

Figura 6 – Um diálogo do jogo Max Payne

2.2.4 Decisões Comerciais

A equipe inicial pretendia formular um jogo com apelo comercial. Por isso, o jogo deveria estar presente no maior número possível de plataformas com expressão comercial disponível. Isso acarretava desenhar o jogo para ser jogado no Facebook, maior rede social da atualidade, e em plataformas móveis. No entanto, para diminuir o escopo do trabalho, foi pensado em desenvolvimento Android[8] como ponto de partida.

A decisão de aumentar o grupo de plataformas alvo acarretou na escolha de um *framework*[9] de desenvolvimento de jogos que desse a capacidade de distribuição para diversas plataformas com o mesmo código fonte.

O *framework* de desenvolvimento escolhido foi a Libgdx (GAMES, 2013). Essa biblioteca tem como os principais alvos de distribuição as plataformas Windows, Linux, Max OS X, Android (+1.5), iOS e Javascript/WebGL. A Libgdx ainda possui uma vasta quantidade de *features* necessárias para o desenvolvimento do ***Os 5 Desprezíveis***, como: câmera ortográfica, *sprite batching* e *caching* de *sprites* automatizado, arquitetura de cenas 2D e atlas de texturas.

A Libgdx é *open source*, sendo que a sua licença de uso é a Apache 2 (FOUNDATION, 2013). Com isso é possível utilizar esse *framework* tanto em projetos comerciais, como em projetos não comerciais. A linguagem de programação utilizada pela Libgdx é Java, a linguagem mais utilizada pelos programadores da equipe inicial.

2.3 Produção

A presente seção apresenta o método de desenvolvimento realizado no processo de construção do jogo ***Os 5 Desprezíveis***. Inicialmente é apresentado o GDD refeito para a nova equipe, seguindo estrutura semelhante ao primeiro. O segundo GDD é apresentado na produção uma vez que o desenvolvimento de alguns *assets*, bem como a produção de componentens, já estava acontecendo. Em seguida, os tópicos de construção do banco de dados e criação de arte são abordados.

2.3.1 Segundo GDD

Devido a dissolução da equipe inicial, não foi possível manter o documento de design original. No entanto, um conceito semelhante foi construído a partir do primeiro GDD do ***Os 5 Desprezíveis***. Isso foi feito para manter aquilo que foi considerado como

[8] http://www.android.com/

[9] Um *framework* (ou arcabouço), em desenvolvimento de *software*, é uma abstração que une códigos comuns entre vários projetos de *software* provendo uma funcionalidade genérica. Um *framework* pode atingir uma funcionalidade específica, por configuração, durante a programação de uma aplicação. Ao contrário das bibliotecas, é o *framework* quem dita o fluxo de controle da aplicação, chamado de Inversão de Controle.

diferencial do jogo, a aleatoriedade e o fator de *replay*. A seguir é apresentado um resumo do segundo GDD.

2.3.1.1 O jogo

Os 5 Desprezíveisé um jogo de investigação que traz para a atualidade o conceito do jogo de tabuleiro Detetive. Seu gráfico é totalmente fotográfico, sendo dado enfoque na ambientação *noir* e no sentimento de tensão em desvendar uma série de assassinatos. Todo o enredo de uma partida é apresentado como um caso à parte, onde o assassino, o cenário, as pistas e outros elementos são gerados aleatoriamente e sofrem influência direta das escolhas feitas pelo jogador através da campanha.

2.3.1.2 *Killer Feature*

A *killer feature* (o diferencial do jogo) é o fato que todo o enredo de cada partida é gerado aleatoriamente, e o jogador é capaz de influenciá-lo diretamente. Isso tudo acontece ainda mantendo a ambientação *noir* que o jogo pretende alcançar.

Outra *feature* que dá destaque ao jogo é o fato que o enredo é montado no início da partida através de um banco de dados. Isso permite que outros caminhos sejam adicionados ao jogo através de atualizações, aumentado ainda mais as possibilidades de enredo para o jogo.

Por fim, o jogo é montado em Atos/Capítulos, onde cada um deles possui uma forma de *gameplay* diferente. Exemplo: um capítulo de investigação; de perseguição; de fuga; de quebra-cabeças; de tribunal; aberto. Além da presença de capítulos inseridos aleatoriamente, também é possível acessar capítulos através de gatilhos: as escolhas do jogador.

2.3.1.3 Controle

O jogador controla um detetive "sem rosto" que é responsável por dar um ponto final em a uma onda de assassinatos. Ao longo da campanha o detetive deve acumular o máximo de pistas e provas para determinar o assassino, enquanto procura se manter vivo. A visão do jogador se dá através de um mapa para navegação ao longo da cidade, e durante a investigação na cena de um crime, ou na exploração de determinado cenário, o jogo é apresentado em primeira pessoa, no estilo *point and click*.

2.3.1.4 Público Alvo

O jogo é voltado para adolescentes e adultos que se interessam por temas criminais e revistas em quadrinhos. Isso se dá devido ao fato que a maior inspiração do jogo são as revistinhas da Vertigo, Dark Horse Comics, principalmente do autor Frank Miller, bem

como os filmes Sin City Fig. (7)., WatchMan, DarkMan. A série CSI também é inspiração para o estilo artístico e para o —gameplay— do ***Os 5 Desprezíveis***.

Figura 7 – SinCity – Clima *Noir*

2.3.1.5 Jogabilidade

O jogo é baseado em uma sequência de capítulos escolhidos aleatoriamente dentre os disponíveis no banco de dados local. Cada capítulo possui alguns eventos próprios e o seu *gameplay* varia ligeiramente. Por exemplo, o capítulo de fuga envolve fazer escolhas em um curto período de tempo. Já os capítulos de pura investigação se baseiam apenas em coletar pistas da cena do crime.

2.3.1.6 Investigação

O capítulo de investigação é a forma mais constante de *gameplay* para o ***Os 5 Desprezíveis***. Ele consiste em procurar pistas em determinada cena de crime. Essa pesquisa se baseia na escolha de navegação ao longo da cena do crime.

Exemplo: O jogador chega a cena do crime (um apartamento) e após uma breve descrição da cena pelo seu ajudante (constante nos principais momentos do jogo), o jogador pode escolher para onde deseja ir, dentre 4 opções apresentadas: quarto, cozinha, banheiro e varanda. Cada cômodo apresenta uma sequência de pistas verdadeiras e falsas que podem indicar o assassino. No entanto, não é permitido ao jogador acessar todos os cômodos: uma pista pode aumentar ou "abrir" novas possibilidade de se encontrar pistas correlatas.

Nos capítulos de investigação é apresentada a principal forma de se concluir um caso com sucesso. As pistas podem levantar dicas para o jogador ou, por serem cabais, incriminar diretamente uma pessoa. É importante notar que nem todas as pistas são verdadeiras, que a condenação de um inocente, bem como a morte do detetive controlado pelo jogador, implicam na falha da campanha.

2.3.2 Planejamento do Escopo

O segundo GDD tinha um escopo mais restrito em relação ao primeiro. No entanto , ele ainda possuía uma complexidade muito maior do que a equipe era capaz de completar dado o período de trabalho estabelecido (cerca de 2 meses). Por isso, diversas decisões foram tomadas para facilitar e viabilizar a construção do ***Os 5 Desprezíveis***, seguindo ao máximo a concepção inicial.

2.3.2.1 Nova tecnologia

Inicialmente era previsto a utilização da Libgdx como a principal ferramenta para construção do ***Os 5 Desprezíveis***. No entanto, esse *framework* possui um grande obstáculo para sua utilização no desenvolvimento: a difícil prototipação. Assim como as bibliotecas SDL, Allegro e outras, a Libgdx não possui uma forma de rápida visualização do que está sendo criado, e nem um programa que funcione como um editor de mapas e cenários. Isso significa que seria quase impraticável a rápida construção de cenas para a avaliação da equipe.

Lembrando da necessidade de se utilizar uma ferramenta que não fugisse do conhecimento de linguagem do programador, e ainda tendo como meta a capacidade de disponibilização do ***Os 5 Desprezíveis***para diversas plataformas, a ferramenta Unity3D foi escolhida para o desenvolvimento.

O Unity3D possui uma ferramenta interna chamada de Editor. Com essa ferramenta é possível navegar pelo mundo do jogo em questão, definir a posição de cada objeto do jogo, inspecionar cada um destes objetos para verificar o seus componentes, ter uma pré-visualização do jogo em questão, visualizar *logs* entre outros. A Fig. (8) apresenta o editor.

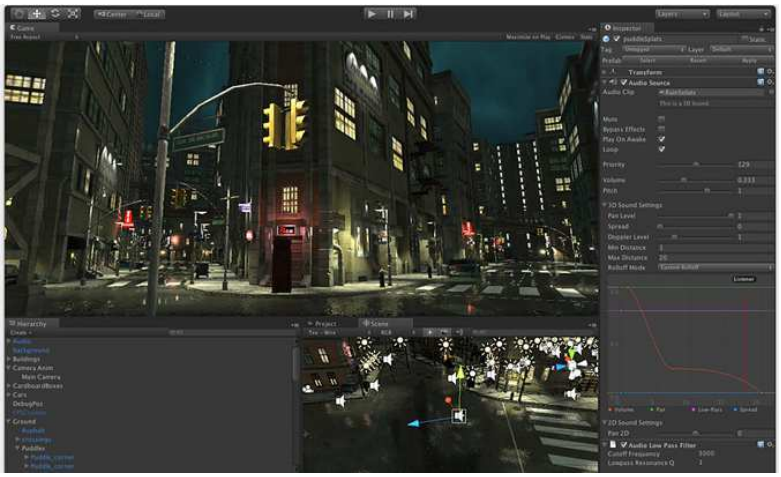

Figura 8 – Visualização do editor do Unity3D

Ao contrário da Libgdx, o Unity3D não é uma ferramenta *open source*, sendo

comercializado pela Unity Technologies. Essa ferramenta é mais que um *framework*, é uma *engine*. Uma *engine* é um conjunto de módulos que compõem o jogo sem estar diretamente relacionados com o comportamento do jogo e nem com o seu ambiente (LEWIS, 2002). Dessa forma, os módulos de áudio, de manipulação de entradas, de visualização 2D e 3D, entre outros, estavam disponíveis apenas com a instalação do Unity3D.

Essa *engine* possui uma grande comunidade, tendo como linguagens de programação o C#, Python Boo e o UnityScript, espécie de dialeto do Javascript, feito exatamente para o Unity3D.

O Unity3D segue uma arquitetura padronizada e possui uma forma diferenciada de programação em relação a Libgdx. A forma em questão é a programação orientada a composição de componente (HASSELBRING, 2006). Esse formato apresenta o conceito de que o comportamento de determinado objeto deve ser composto por diversos componentes. Dessa forma, se for necessário mudar o comportamento de determinado objeto, basta apenas retirar o componente que causa o comportamento que se deseja eliminar ou substituir o componente por outro com o comportamento desejado.

A composição de componentes pode oferecer grandes vantagens para o desenvolvimento de jogos. Algumas dessas vantagens são: reusabilidade de comportamentos, fácil customização, adaptação e fácil manutenção de um objeto de jogo.

2.3.2.2 Cortes no roteiro

Para construção do ***Os 5 Desprezíveis***, alguns itens foram priorizados. A primeira parte a ser implementada foi o capítulo de investigação, uma vez que esse capítulo representa a maior parte da jogabilidade necessária para concluir o jogo. O segundo capítulo a ser implementado foi o de acusação.

Os demais capítulos foram adiados para uma próxima versão. Isso aconteceu devido a distância na forma de jogabilidade apresentado por eles. O capítulo de fuga é completamente diferente do investigação e do caça. Essa variabilidade tornaria o desenvolvimento muito mais longo, exigindo um tempo muito maior que a data limite para a conclusão do trabalho permitia. Dessa forma, para diminuir riscos e manter a entrega do ***Os 5 Desprezíveis***, esses capítulos foram excluídos do desenvolvimento.

2.3.3 Banco de dados

A randomicidade dos casos é fator determinante do jogo. Para tanto, era necessário construir um sistema capaz de trazer a sensação de randomicidade para cada caso. No entanto, não faz sentido utilizar números realmente randômicos (SCHINDLER, 2001), uma vez que para o jogo não existe perdas em como os números são gerados, assim como não há prejuízo para o jogador.

A saída encontrada foi construir um banco de dados com as entradas necessárias do jogo, e dentro desse banco randomizar os itens para cada caso. Dessa forma, quanto maior a quantidade de itens dentro do banco de dados, maior a sensação de randomicidade. Por exemplo, cada suspeito deve possuir 3 itens de quadro clinico (depressivo ou não, ébrio contumaz ou ex-ébrio contumaz, usuário de drogas ou ex-usuário), sem que o jogador saiba dessa característica, é possível trazer a sensação de randomicidade com diversas descrições para o mesmo quadro. O ébrio contumaz pode ser ter a descrição como: "Alcoólatra social. Bebe para esquecer dos problemas" ou "Alcoólatra. Sempre que vai a uma festa, sai carregado pelos amigos".

Fora a descrição do estado fornecer a maior variedade, a forma com que o ***Os 5 Desprezíveis***foi construído permite a inclusão de: novos tipos, descrições de qualquer tipo de factoids (quadro clínico, *hobbys* e profissões), *non-player character* (NPC, personagens não jogáveis), armas, áreas e lugares. Isso facilita muito a criação de novos conteúdos sem que seja exigido uma grande alteração na lógica do ***Os 5 Desprezíveis***, sendo até possível lançar um jogo completamente diferente apenas mudando o banco de dados.

2.3.4 Criação de diálogos e arte

As descrições e falas do ajudante do detetive são de suma importância para a jogabilidade do ***Os 5 Desprezíveis***. Dessa forma, o trabalho do redator deveria ser facilitado. Para isso, foi utilizado uma ferramenta livre com o intuito de manipular o banco de dados de maneira intuitiva. A ferramenta escolhida foi o SqlStudio (STUDIO, 2013).

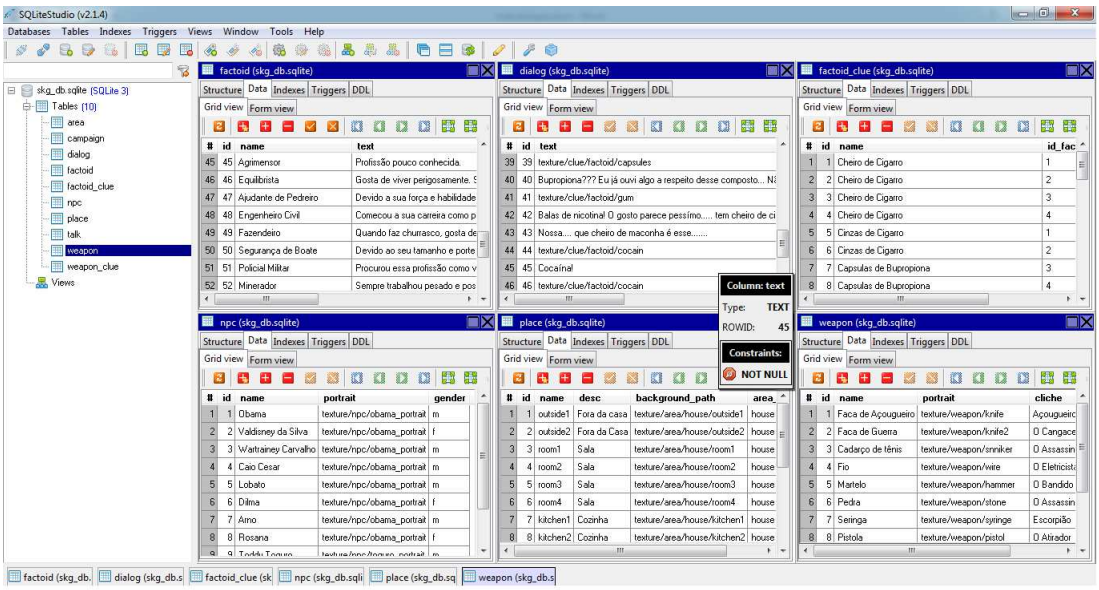

Figura 9 – SQL Studio, estúdio para sqlite

Não tendo nenhum integrante na equipe voltado apenas para a produção de arte gráfica, a equipe escolheu duas abordagens diferentes. A primeira foi a utilização de gráficos foto realistas. Ou seja, todo cenário, personagem ou qualquer outro *asset* no jogo é apresentado por uma foto. Dessa forma, o maior obstáculo para representar determinado objeto é encontrar a foto ideal.

Para encontrar todas fotos em tempo hábil, a equipe foi "aberta" para entrada de colaboradores, que teriam seus nomes publicados nos créditos. O mesmo foi feito para as fotos tiradas dos NPCs do ***Os 5 Desprezíveis***. Assim se tornou altamente viável selecionar as fotos enviadas sem o grande trabalho de buscar pelas mesmas, restando apenas as tarefas de selecionar fotos e filtrá-las para o formato do jogo. As figuras Fig. (10) e Fig. (11) mostram uma imagem enviada e o estado dela depois dos filtros, respectivamente.

Figura 10 – Figura enviada por colaboradores

Figura 11 – Figura filtrada e indexada no banco de dados

A trilha sonora foi composta apenas com sons que possuíam a licença Creative Commons (COMMONS, 2013) em que a comercialização fosse permitida. Todas as músi-

cas foram selecionadas do artista Kevin MacLeod[10], que apenas pede a sua referência no trabalho produzido.

2.3.5 Controle e manipulação do código

Para manter cópias de segurança, bem como facilitar as linhas de desenvolvimento do ***Os 5 Desprezíveis***, foi necessário a adoção de um sistema que facilitasse o controle entre diversas versões de código. A ferramenta escolhida para tanto foi o GIT, com o uso da hospedagem no GitHub[11]. Para melhor aproveitamento da ferramenta, foram produzidos 2 repositórios: um público, contendo o código de *script* e outro privado contendo apenas produção artística. A escolha pela ferramenta e hospedagem, além de contribuir para gestão de configuração do jogo, serviu como insumo para o controle da carga de trabalho e avaliação do objeto de trabalho.

A hospedagem GitHub produz diversos gráficos com base nos *commits* realizados e com base no próprio repositório em questão. Entre os gráficos está o de frequência de código Fig. (12), que indica a quantidade de linhas de código que foram adicionadas e quantas foram removidas por semana.

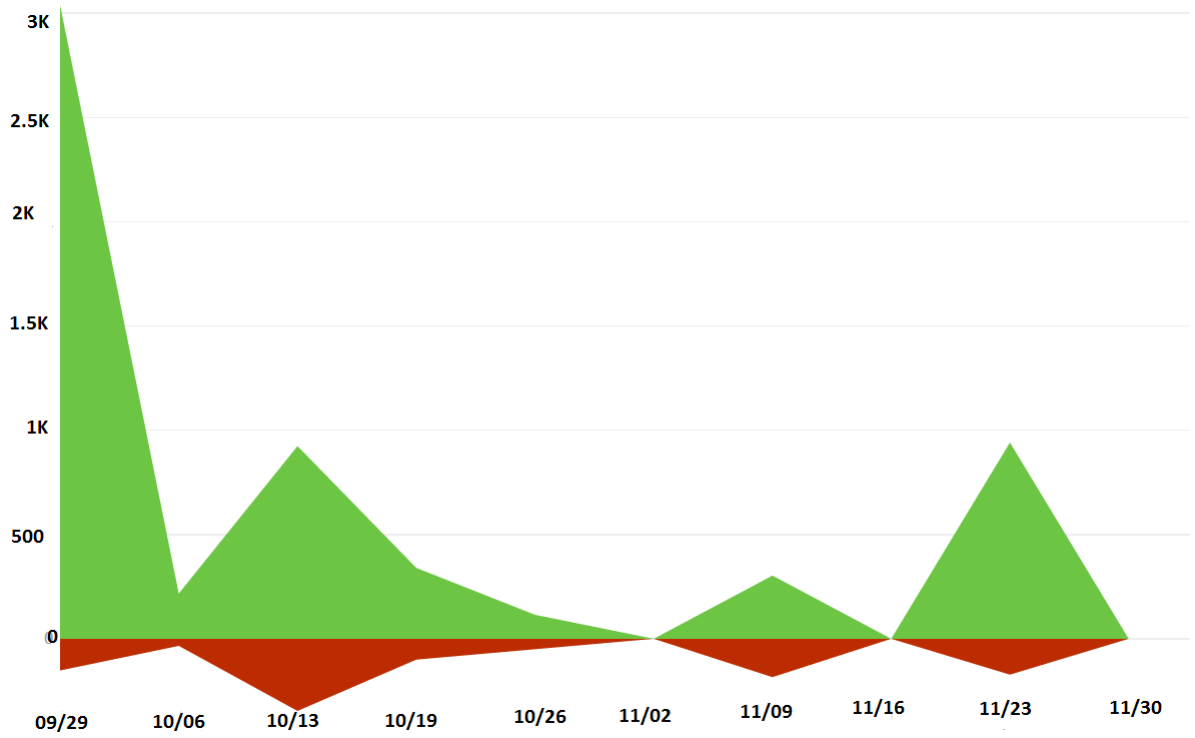

Figura 12 – Linhas de código adicionadas e removidas

Com o gráfico de frequência de código é possível ver em quais momentos houve a maior produção de código *script*. Esses momentos coincidiram com o início do desenvolvi-

[10] http://incompetech.com/
[11] https://github.com/eusyar/mvc

mento e com o final. Como o GitHub avalia apenas a frequência de código, não é possível utilizar essa ferramenta para medir a quantidade de arte produzida.

2.4 Pós-Produção

A pós-produção foi o momento em que o jogo entrou fase de testes alfa. Os testes alpha foram conduzidos junto a jogadores, na presença da equipe. Nesse momento, pontos de melhoria e *bugs* foram anotados pela equipe. Foi procurado seguir a sequência de perguntas:

- O jogo estimulou você?
- Você sentiu estimulado para terminar o jogo?
- Ficou claro o que era preciso fazer?
- O jogo foi muito difícil?
- Deveria haver um modo de mais difícil/fácil do jogo?
- A quantidade de tutorial foi satisfatória?
- O jogo foi recompensador?
- Os itens estavam disponibilizados corretamente?

Com o jogo em estado alfa, as *cutscenes*[12] foram refeitas para tornar mais imersivo parte do jogo que não tiveram tanto destaque na produção. As mais reforçadas foram as que serviriam de tutorial para o jogador.

Atividades comuns de transição, como a confecção de um instalador para o jogo, testes beta e criação de um *trailer* para o jogo não foram executadas.

Confecção de executáveis é automatizada dentro do Unity3D. Ele é responsável por criar *builds* para diversos tipos de alvos e sistemas operacionais e como não foram executados testes beta, a equipe decidiu por manter apenas os executáveis ao invés de instaladores completos.

Os testes beta não foram executados porque é necessário uma base de usuários maior e segurança da equipe em disponibilizar o software para esse grupo. Devido ao resultado dos testes alfa foi escolhido não abrir o jogo para um grupo maior, focar apenas nos erros apontados e nas *features requests*.

Usualmente, a pós-produção tem como principal atividade a preparação e comercialização do jogo. Contudo essa etapa não foi executada nesse trabalho.

[12] Cenas de diálogo sem jogabilidade propriamente dita

3 Resultados e Conclusões

3.1 Resultados

No final do desenvolvimento foi criado um jogo completo em sua proposta. Apenas o capítulo de investigação e o de acusação foram criados, deixando os demais capítulos para um próxima *release*.

3.1.1 Testes Alfa

Os testes alfa ocorreram nas últimas semanas de desenvolvimento, com pessoas não ligadas diretamente ao jogo, principalmente colaboradores. O intuito dos testes alfa eram encontrar *bugs* e encontrar possíveis pontos de melhoria através de um questionário e entrevistas.

No entanto, os testes foram realizados com um grupo muito reduzido de colaboradores, apenas 4. Isso se provou como uma péssima escolha, uma vez que os resultados da pesquisa tiveram pouquíssima divergência tendo apenas duas perguntas com respostas diferentes. Apenas nas perguntas "Ficou claro o que era preciso fazer?" e "Os itens estavam disponibilizados corretamente" houve 50% de respostas negativas.

Ficou claro que o questionário realizado foi muito curto para o teste alfa. Era preciso avaliar outros itens como a satisfação com as músicas, fluxo de jogo, *feedback* e outros. No final, apenas uma parte muito pequena da metodologia *GameFlow* foi executada, apenas o questionário para teste alfa cobrindo apenas os itens de: o jogo; concentração; desafio e controle.

Não houve classificação de *bugs*, nem a utilização de ferramentas para acompanhamento de defeitos, como BugZilla[1], no desenvolvimento. A motivação para essa escolha foi o tamanho da equipe. No entanto, isso foi considerado um erro, uma vez que a maior parte dos erros encontrados no teste alfa já haviam sido reconhecidos durante o desenvolvimento.

Depois de percorrer o jogo avaliando os *bugs*, os colaboradores ajudaram indicando os principais pontos de melhoria. Isso levou aos principais pedidos de atualização para próximas versões:

- O jogo precisa de mais capítulos e mais episódios de investigação;
- Mais pistas, mesmo que não sejam conclusivas;

[1] http://www.bugzilla.org/

- Mais opções de manipulação da pista;
- Revisar as pistas;
- Mais diálogos e mais personagens;
- Escolher quais pistas seriam utilizadas no júri;
- Uma estrutura maior de navegação, com delegacias, laboratórios;
- Implementação de eventos.

Outra *feature* muito requisita foi a implementação do conceito evolução do personagem, através de aquisição de novas habilidades, características e itens úteis para evolução do jogo, a exemplo de jogos como ***Megaman***.

3.1.2 Arte

Devido as escolhas em relação a forma de tratamento de *assets*, não houve uma direção de arte. Isso impactou na divergência clara entre o clima planejado e o alcançado. O GDD previa uma ambientação muito *noir*, no entanto o que foi alcançado foi uma espécie de humor sombrio, lembrando o filme *Who Framed Roger Rabbit*[2].

A ausência de um responsável por conteúdo artístico alterou a orientação do jogo. Notou-se que é preferível que a direção artística seja executada por alguém com mais de uma responsabilidade do que não o ser. A presença de um diretor artístico mostrou-se fundamental e necessária para manter o planejamento do GDD no que tange a sua identidade visual.

Boa parte da arte foi feita por colaboradores que enviaram fotos, para participarem como NPCs ou para ajudar a composição das cenas, o que gerou uma falta de consonância entre algumas cenas e personagens, fato evidenciado pelas Figuras 13 e 14.

Uma vez que foi mais fácil permitir a alteração de conceito e conseguir a quantidade de *assets* necessário do que orientar exatamente como deveriam ser produzidas as fotos. O tamanho dos arquivos resultantes aumentou muito se comparado ao esperado. Os mais de 50 arquivos de imagens totalizaram cerca de 30MB. Contudo, a maior parte do tamanho do jogo não foi resultado das imagens e sim do áudio.

Os arquivos de áudio não sofreram tratamento: eles estão presentes dentro do jogo da mesma forma com que foram adquiridos. Esse se mostrou um erro de construção do jogo do ponto de vista da *build* final, ocupando cerca de 72% do tamanho da pasta de *assets*, 80mb.

[2] http://www.imdb.com/title/tt0096438/

Figura 13 – Personagem segundo o conceito inicial

Figura 14 – Personagem e apresentação segundo colaboradores

3.1.3 Jogabilidade

De forma geral, a jogabilidade pretendida no jogo não foi alcançada. Isso se deve principalmente ao fato que os demais capítulos do jogo (fuga, caça, etc) não foram implementados. No entanto, outros itens foram repensados para o capítulo de investigação, o que mudou a jogabilidade.

Na versão final, o jogo facilita o encontro de pistas. Quando o jogador se aproxima de uma pista, ele começa a brilhar, mesmo sem que o *mouse* esteja logo em cima dela. Isso foi feito para evitar que o jogador se sinta frustrado por não encontrar nenhuma pista e que ele clique incessantemente por toda cena em vão. A figura 15 mostra um caso onde

a pista é indicada à curta distância.

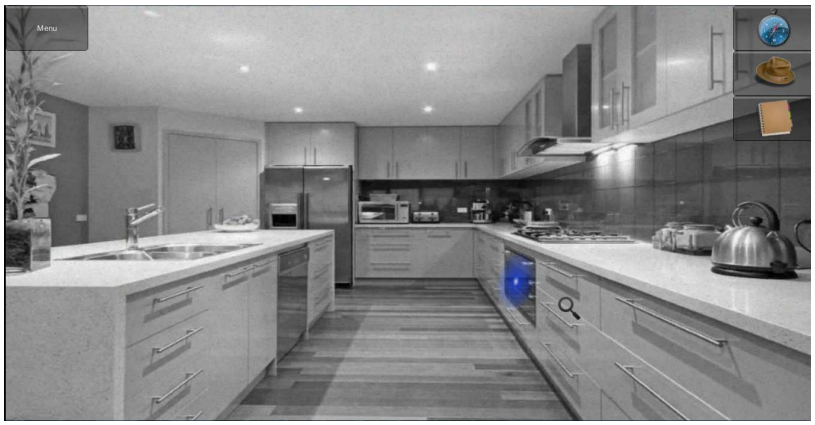

Figura 15 – A lupa indica a posição do *mouse*, o círculo azul a pista

3.2 Utilização de ferramentas

Ainda que a escolha do Unity 3D como tecnologia para desenvolvimento tenha sido uma solução viável, a falta de perícia da ferramenta causou um grande retrabalho. Sem a familiaridade com a orientação a componentes, boa parte do código se tornou redundante e ineficiente.

Boa parte dos componentes possui uma alta dependência de outros componentes, ou de até mesmo de objetos definidos dentro de cena. Isso vai contra a ideia de reutilização de componentes, sendo que boa parte dos componentes só pode ser utilizada uma vez em ***Os 5 Desprezíveis***.

Um caso claro é o *script* ContextMenu, onde não existe parametrização de entrada para o tamanho, caminho dos ícones e a solução está atrelada a saída para determinada cena, conforme o Código **??**

```
1 if (GUI.Button (Rect (0,placesArray.Count*this.rectNavViewPort.height/
2 (placesArray.Count+1) + 10, this.rectNavViewPort.width,
3 this.rectNavViewPort.height/(placesArray.Count+1)),"Sair da Area")){
4         this.showNavBox = false;
5         Application.LoadLevel("map");
6 }
```

Listing 3.1 – Codigo para menu de contexto

Grande parte dos *assests* eram indexados no banco de dados, por isso foi necessário forçar a sua compressão junto ao jogo construído. Esta não é uma pratica aconselhada pela comunidade do Unity 3D, pois força o acompanhamento da pasta de *assests* junto ao *build* do jogo.

Devido a essa aproximação errônea, o tamanho do jogo cresceu de maneira incontrolada. Quando o jogo é criado utilizando a indexação do próprio Unity3D, apenas os recursos utilizados são compactados e adicionados a *build*, diminuindo o tamanho final do executável. Como ***Os 5 Desprezíveis***foi feito com imagens indexadas através do banco de dados, o arquivo executável, mais a pasta de recursos (*assets*) e a pasta de data compilada somam 240MB de espaço em disco.

3.3 Conclusão

Esse trabalho passou por boa parte do ciclo de vida do desenvolvimento de um jogo. Tendo como resultado um jogo completo. No entanto, devido a sua forma de desenvolvimento ter sofrido algumas falhas artísticas e técnicas, o ***Os 5 Desprezíveis***provavelmente não será disponibilizado a curto prazo.

A utilização de métodos de Engenharia de Software estiveram presentes em todo o processo de desenvolvimento. Eles foram grandes aliados, principalmente para mitigar riscos do projeto. No entanto, vários aspectos foram subutilizados, como as cerimônias do Scrum, que deixaram de ser utilizadas devido ao tamanho da equipe. Isso demonstra uma grande dificuldade de executar todas as cerimônias na realidade de um trabalho de conclusão de curso, onde, geralmente, não existe mais de 2 duas pessoas trabalho no projeto.

A construção do GDD foi muito útil para manter um foco no desenvolvimento. No entanto, mesmo tentando evitar construir o GDD e não atualiza-lo, devido ao tamanho da equipe, ele foi sendo usado apenas como uma referência.

Ao longo do desenvolvimento, ficou claro a altíssima dependência da produção artística aliada a codificação, ainda que a codificação tenha sido feita de maneira quase paralela a criação de conteúdo artistico. Mesmo que tenha sido utilizado de meios diferenciados de se produzir conteúdo artístico, a ausência de especialistas na área fizeram com que o jogo ficasse aquém do esperado. A maior do tempo de desenvolvimento ficou para o acabamento artístico. A Figura 12 mostra gargalos de desenvolvimento, nesses pontos houve exatamente o maior esforço para a produção artisitica.

O método *GameFlow* foi utilizado durante o desenvolvimento servindo como um guia para a prototipação, diminuindo o retrabalho nessa fase e orientando a equipe a perseguir *features* que inicialmente pareciam ter menor importância, no entanto, quando ele foi utilizado para os testes alfa, o resultado não foi satisfatório. Isso aconteceu principalmente devido a falhas na construção do questionário e na seleção de colaboradores para realizar os testes. A falha na execução mostra a necessidade de um integrante da equipe voltado para os testes, e ainda mostra que a elaboração dos teste deve ser feita ainda na produção.

No final foi possível exercitar vários conceitos ligados ao curso de Engenharia de Software. São muitas as possibilidades de pesquisa na produção de jogos, como métricas para tamanho e preço e avaliação de qualidade final e qualidade no processo.

Referências

ABRAGAMES. A indústria brasileira de jogos eletrônicos um mapeamento do crescimento do setor nos úl6mos 4 anos. 2008. Disponível em: <http://www.abragames-.org/wp-content/uploads/2013/04/Abragames-Pesquisa_2008.pdf>. Citado na página 22.

ASSOCIATION, E. S. et al. Essential facts about the computer and video game industry. v. 1, 2013. Citado na página 21.

BAER, R. H.; BURNHAM, V. *Supercade: A visual history of the videogame age, 1971-1984*. [S.l.]: MIT Press, 2001. Citado na página 24.

BALDWIN, M.; CONSULTING, B. Game design document outline. 2005. Citado na página 28.

BECK, K.; ANDRES, C. *Extreme programming explained: embrace change*. [S.l.]: Addison-Wesley Professional, 2004. Citado na página 34.

BECK, K. e. a. *Agile Manifest*. 2013. Disponível em: <http://agilemanifesto.org/>. Citado na página 32.

BETHKE, E. *Game Development and Production*. [S.l.: s.n.], 2003. Citado 2 vezes nas páginas 21 e 31.

BISWAS, A.; SINGH, J. Software engineering challenges in new media applications. 2006. Citado 2 vezes nas páginas 27 e 31.

COMMONS, C. *Licença de Uso*. 2013. Disponível em: <http://creativecommons.org-/licenses/by-sa/4.0/legalcode>. Citado na página 50.

DEMACHY, T. *Extreme Game Development: Right on Time, Every Time*. 2013. Disponível em: <http://www.gamasutra.com/resource_guide/20030714/demachy_01-.shtml>. Citado 3 vezes nas páginas 31, 32 e 34.

FLOOD, K. *Game Unified Process*. 2013. Disponível em: <http://www.gamasutra.com-/view/feature/2827/extreme_game_development_right_on_.php>. Citado 4 vezes nas páginas 31, 32, 33 e 35.

FOUNDATION, A. S. *Licença de Uso*. 2013. Disponível em: <http://www.apache.org-/licenses/LICENSE-2.0.html>. Citado na página 44.

GAMES, B. L. *Feature List*. 2013. Disponível em: <http://libgdx.badlogicgames.com-/features.html>. Citado na página 44.

GODOY ANDRé, e. B. F. Game-scrum: An approach to agile game development. In: *SBC - Proceedings of SBGames 2010*. [S.l.: s.n.], 2010. Citado 3 vezes nas páginas 27, 31 e 32.

HAGIU, A.; HALABURDA, H. *Responding to the Wii?* [S.l.]: Harvard Business School, 2009. Citado 3 vezes nas páginas 23, 26 e 27.

HASSELBRING, W. Component-based software engineering. In: *International Journal of Software Engineering and Knowledge Engineering*. [S.l.: s.n.], 2006. v. 1. Citado na página 48.

III, H. W. P. The facts on home video games: From the man who plays games for a living. 1989. Citado na página 26.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. E. *The unified software development process-the complete guide to the unified process from the original designers*. [S.l.]: Addison-Wesley, 1999. Citado 2 vezes nas páginas 32 e 35.

KEITH, C. *Kanban for video game development*. 2013. Disponível em: <http://www-.infoq.com/presentations/kanban-video-game-dev>. Citado na página 37.

KENT, S. *The Ultimate History of Video Games*. [S.l.: s.n.], 2001. Citado 4 vezes nas páginas 23, 24, 25 e 26.

LAPLANTE, P. A.; NEILL, C. J. The demise of the waterfall model is imminent. *Queue*, ACM, v. 1, n. 10, p. 10, 2004. Citado na página 32.

LEWIS, J. J. M. *Game Engines in Scientific Research*. [S.l.: s.n.], 2002. 27-31 p. Citado na página 48.

MANKER, J. Game design prototyping. In: *Games and Innovation Research Seminar 2011 Working Papers*. [S.l.: s.n.], 2011. p. 41. Citado na página 28.

NEVELSTEEN, K.; GAYOSO, S. Gdd as a communication medium. In: *Games and Innovation Research Seminar 2011 Working Papers*. [S.l.: s.n.], 2011. p. 53. Citado na página 42.

NINTENDO. *Informações sobre o histórico da empresa*. 2013. Disponível em: <http://www.nintendo.com/corp/history.jsp>. Citado na página 26.

OLIVER, R. L. N. M. B. Exploring the concept of media enjoyment: An introduction to the special issue. 2004. Citado na página 28.

PETRILLO, F. et al. Houston, we have a problem...: a survey of actual problems in computer games development. In: ACM. *Proceedings of the 2008 ACM symposium on Applied computing*. [S.l.], 2008. p. 707–711. Citado na página 21.

RABIN, S. *Introduction to game development*. [S.l.]: CengageBrain. com, 2010. Citado na página 40.

ROGERS, S. *Level Up!: The Guide to Great Video Game Design*. [S.l.]: Wiley. com, 2010. Citado na página 30.

SANTOS, A. R. Agile game process : Metodologia Ágil para projetos de advergames. 2006. Disponível em: <http://www.cin.ufpe.br/~tg/2006-1/arsa.pdf>. Citado 2 vezes nas páginas 21 e 32.

SCHINDLER, W. Efficient online tests for true random number generators. In: SPRINGER. *Cryptographic Hardware and Embedded Systems—CHES 2001*. [S.l.], 2001. p. 103–117. Citado na página 48.

SQUIRE, K. Video games in education. *Int. J. Intell. Games & Simulation*, v. 2, n. 1, p. 49–62, 2003. Citado na página 21.

STUDIO, S. *Feature List*. 2013. Disponível em: <http://sqlitestudio.pl/>. Citado na página 49.

SWEETSER, P. Gameflow: a model for evaluating player enjoyment in games. 2005. Citado na página 28.

TANG, R. Rapid application development. 1997. Citado na página 32.